**Tareas a realizar:**

1. Elaborar código en C++ para la lectura de datos necesarios para el modelo

* Asumimos que todos los datos están en archivos de texto separados por comas. Para las pruebas crearemos archivos pequeños, más adelante implementaremos código en Python para generar estos archivos a partir de los archivos Excel de la EPMTP.
* Leer un archivo terminales.csv que debe contener los campos: id\_terminal (int), nombre (string), num\_buses (int). Construir con esta información un arreglo terminales, cuyos elementos son objetos de la clase terminal, que representan a cada uno de los terminales. Cada objeto debe tener los campos señalados.
* Leer un archivo viajes.csv que debe contener los campos: id\_viaje (int), id\_terminal\_salida (int), id\_terminal\_llegada (int), tiempo\_inicio (int), tiempo\_fin (int), nom\_circuito (string). Construir con esta información un arreglo viajes, cuyos elementos son objetos de la clase viaje, que representan cada uno de los viajes. Cada objeto debe tener los campos señalados. Es importante que los viajes estén ordenados por tiempo de inicio.

1. Construir el archivo de encabezado parametros.h en el que se usarán instrucciones #define para definir todos los parámetros descritos al final de este documento. Tanto en el código como en este documento identificamos a los parámetros escribiendo TODO\_EN\_MAYUSCULA
2. Construir el multigrafo
   * El grafo se representará como un objeto de la clase multigrafo\_vsp, definida como se indica a continuación
   * Definir las variables: n (número de viajes) y d (número de depósitos o terminales)
   * El grafo contiene como nodos los números {1, …, n+2d}, de la siguiente manera: los números {1, …, d} identifican los depósitos de salida, los números {d+1, …, d+n} identifican los viajes, los números {n+d+1, …, n+2d} identifican los depósitos de llegada. Para k \in {1,…,d} el nodo k corresponde a terminales[k-1]. Para i \in {d+1,…,n+d} el nodo i corresponde a viajes[i-d-1]. Para k \in {n+d+1,…,n+2d} el nodo k corresponde a terminales[k-n-d-1].
   * Los arcos se representan por tripletas de enteros (i,j,k), donde i es el nodo de salida, j el nodo de llegada y k el tipo.
   * Crear dos vectores de listas arcos\_sal y arcos\_ent para almacenar las listas de arcos entrantes y salientes de cada nodo. Los arcos se almacenarán por duplicado en estas dos listas. Por ejemplo, el arco (i,j,k) aparecerá como un elemento de las listas arcos\_sal[i] y arcos\_ent[j].
   * Agregar los arcos entrantes y salientes del depósito: Para i \in {1,…,n}, consultar el depósito de salida k\_0 y el depósito de llegada k\_1 de viajes[i-1]. Agregar los arcos (k\_0, d+i, k\_0) y (d+i, k\_1+d+n, k\_1) a donde corresponda en las listas de arcos entrantes y arcos salientes.
   * Para i \in {1,…,n}, para j \in {i+1,…,n} , j es compatible con i, agregar los arcos (d+i,d+j, k) para k \in 1,…,d, a las listas de arcos entrantes y arcos salientes. Para que j sea compatible con i deben cumplirse dos condiciones:
     1. Terminal de salida de j es terminal de llegada de i
     2. Tiempo de inicio de j >= Tiempo fin de i + T\_MIN\_ESPERA
   * Definir los costos de los arcos como un unordered\_map<arco, double> c\_primal, de tal forma que c\_primal[i,j,k] almacene el costo del arco (i,j,k). Los costos son iguales a:
     1. COSTO\_FIJO\_RUTA para los arcos que salen de un depósito
     2. (tiempo inicio j – tiempo fin i) \* FACTOR\_COSTO\_TIEMPO\_MUERTO para los arcos que conectan un viaje i con un viaje j
     3. Cero para los arcos que llegan a un depósito
3. Construir el PL inicial
   * Construir un PL que tiene una variable (columna) por cada viaje y una variable (columna) por cada depósito. Cada una de estas columnas cubre solamente el viaje o el depósito respectivo, es decir, las restricciones son:
     1. x\_i = 1, para todo viaje i
     2. x\_k <= num\_buses(k), para todo depósito k
   * Los costos de cada una de estas columnas son iguales a COSTO\_COLUMNA\_ARTIFICIAL
   * Todas las variables son no negativas y de tipo continuo
4. Resolver el PL inicial, consultar los precios duales de los viajes y depósitos y ajustar los costos del multigrafo
   * Resolver el LP y consultar los precios duales alpha[i] (para las restricciones de los viajes) y beta[k] (para las restricciones de los depósitos).
   * Definir los costos modificados del grafo como un unordered\_map<arco, double> c, cuyos valores están dados por:
     1. c[i,j,k] = c\_primal[i,j,k] – alpha[i], si i es un viaje
     2. c[i,j,k] = c\_primal[i,j,k] + beta[i], si i es un depósito

**Lista de parámetros:**

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Descripción** | **Valor por defecto** |
| T\_MIN\_ESPERA | Tiempo de espera mínimo entre dos viajes consecutivos | 5 |
| COSTO\_FIJO\_RUTA | Costo fijo por el uso de un vehículo en una ruta | *Por determinar* |
| FACTOR\_COSTO\_TIEMPO\_MUERTO | Costo de cada minuto de tiempo muerto en una ruta | 1 |
| COSTO\_COLUMNA\_ARTIFICIAL | Costo de cada columna artificial utilizada para inicializar el LP | 1e10 |
|  |  |  |
|  |  |  |